1.Introducción al Modelo Relacional.

1.1 ¿Qué es un Modelo ?.

Cuando en teoría de diseño de bases de datos se emplea el término "modelo" , esto no tiene el mismo significado que en Lógica .
En Lógica por "modelo" se entiende una interpretación en la que se satisfacen (se evalúan como ciertas) las fórmulas de una teoría , lo que asegura su consistencia .
En Bases de Datos , un Modelo es un formalismo que nos permite representar la realidad y , más concretamente , aquélla parte de la realidad que nos interesa (Una Empresa , una Universidad , una Biblioteca , el catálogo de las especies protegidas en un determinado país...).
Las partes fundamentales de un Modelo , son :

- Estructura de datos
- Restricciones
- Operadores asociados.

1.2 Estructura de datos en el Modelo Relacional.

El Modelo Relacional fue propuesto por E.Codd en 1970 (E. Codd era entonces un investigador del Centro de IBM en San José (California) y publicó su propuesta en un artículo fundamental que obtuvo el ACM Award correspondiente al Congreso VLDB de 1970) .
La estructura subyacente básica es la **relación** , entendida en su acepción matemática básica : Un subconjunto de un producto cartesiano de conjuntos .

Cuando se pretende describir una parcela de la realidad mediante el formalismo relacional , el primer paso es discernir los **atributos** presentes en el problema .
Un atributo es un ítem elemental de información , en el sentido de no poder desglosarse en componentes más simples.
Los atributos son los nombres que damos a las propiedades de los objetos acerca de los cuales se va a guardar información .

Supongamos que se trata de crear una Base de Datos para una Empresa determinada.
Los datos considerados relevantes por cada empleado , son :

- DNI
- Nombre
- Dirección
- Teléfono
- E-mail
- Puesto
- Antigüedad en la Empresa
- Tipo de salario que se aplica

Estos son por tanto los atributos de empleado relevantes para el problema .

Llamamos **dominio** al conjunto en el cual un atributo toma sus posibles valores .
En nuestro ejemplo :
Dom (DNI) = { cadenas de caracteres alfanuméricos de 9 elementos }
Dom (Nombre)={ cadenas de caracteres alfabéticos de un número máximo de elementos }
Dom (Teléfono)={cadenas de caracteres numéricos de 9 elementos}
Etc…….

En un mismo problema no puede haber dos atributos diferentes con el mismo nombre , pero sí que dos o más atributos diferentes pueden tener dominios idénticos .

En el ejemplo , cada empleado queda descrito por los correspondientes valores de los atributos anteriormente enunciados . Cada empleado se corresponde con una **n-tupla** de valores y el conjunto de las tuplas correspondientes a todos los empleados de la Empresa es la **relación** que podemos denominar “Empleados” por motivos obvios .
Las relaciones se representan en forma tabular , con las columnas etiquetadas por los atributos correspondientes .
Al número de columnas se le denomina “grado” u “orden” de la relación y al número de columnas se le denomina “cardinalidad”.

Empleados

DNI	NOMBRE	DIRECCION	TELEFONO	E_MAIL	PUESTO	ANT.	SAL.
1023R	M.Fernández	Luna 25 Madrid 28025	913367391	mf@pruebas.es	Sales-manager	3	30.000

El ejemplo muestra el contenido de una de las tuplas de la relación “Empleados”. Esta tupla es un elemento del producto cartesiano de los dominios de los diferentes atributos , de modo que podemos afirmar:

Empleados $\subseteq$ Dom(DNI)xDom(NOMBRE)x…xDom(SALARIO)

Una relación es un subconjunto del producto cartesiano de los dominios de los atributos sobre los cuales está definida. Sus elementos son las tuplas .

1.3 Operadores del modelo.

Hay dos tipos de lenguajes relacionales : Los basados en el Cálculo Relacional y los basados en el Algebra Relacional .

El Cálculo Relacional es un subconjunto del Cálculo de predicados de primer orden (sin símbolo de negación y sin símbolos de función) , que tiene la propiedad (demostrada por Codd) de ser relacionalmente completo , es decir : Cualquier información presente en la Base puede ser obtenida formulando una interrogación en términos de Cálculo relacional.
Las interrogaciones son de la forma < meta / condición > , siendo la condición una fórmula cerrada del cálculo .
Estos lenguajes pueden ser orientados a tuplas (ej. : DSL ALPHA) u orientados a dominios (ej. : QBE) , según el tipo de variable que empleen . En todo caso , son lenguajes de alto nivel , no procedimentales , ya que descargan al usuario de la responsabilidad de planificar sus accesos con criterio de optimización de los tiempos de respuesta / ejecución .
La desventaja es que requieren de una familiaridad con la Lógica que no suele tener el usuario “normal”.
El ámbito en que estos lenguajes encuentran su campo natural de aplicación son las Bases de Datos Deductivas , en las que la Lógica es el marco adecuado de descripción uniforme de hechos y reglas , y la repuesta a una interrogación es un caso particular de demostración de teoremas .

El Algebra es un conjunto de operadores que admiten como operandos relaciones y devuelven como resultado relaciones .
Es una ampliación del Algebra convencional de conjuntos (teniendo en cuenta que el elemento de una relación es la tupla) , añadiendo algunos operadores específicos para relaciones .
Es fácil demostrar que todo operador del Algebra puede expresarse en términos del Cálculo , con lo que la completitud relacional de aquélla está también asegurada .
El Algebra es procedimental , así que es importante considerar algunas heurísticas de optimización para minimizar tiempos de respuesta .
El estándar de los lenguajes relacionales , el SQL , está basado en Algebra .

1.4 Restricciones del modelo .

Distinguiremos las **básicas** y las **dependencias .**

Restricciones básicas :

- **Integridad de entidad .**
- **Integridad referencial .**

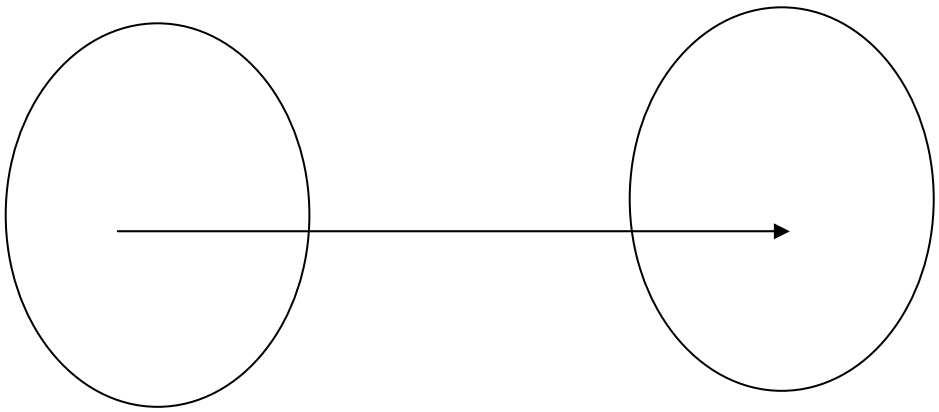
Describiremos estas restricciones una vez introducido formalmente el concepto de **clave** .

Dependencias :

- **Dependencias funcionales**
- **Dependencias multivaluadas**
- **Dependencias de unión**
- **Dependencias generalizadas (proporcionan el marco para la descripción uniforme de los tipos anteriores) .**

Las dependencias funcionales expresan propiedades inherentes a los datos . **Un descriptor (conjunto de atributos) Y depende funcionalmente de otro descriptor X si el valor de X determina unívocamente el de Y y se expresa así : $X \rightarrow Y$ (X implica Y) .**

Dom (X) **Dom (Y)**



Una clave es un descriptor mínimo (no redundante) del cual dependen funcionalmente los demás atributos de una relación (en lo sucesivo , para hacer referencia a una estructura de relación con independencia de su contenido , que , en general , variará con el tiempo , usaremos el término " esquema de relación " , o, si no hay ambigüedad , simplemente "esquema") .

En el esquema **Empleados ,** es inmediato comprobar que **DNI** $\rightarrow$ **NOMBRE , DNI** $\rightarrow$ **DIRECCION …DNI** $\rightarrow$ **SALARIO ,** de modo que **DNI** es clave del esquema.

En un esquema puede haber más de una clave . Los atributos que forman parte de alguna clave se llaman principales y los demás no principales .

La **integridad de entidad** establece que ningún atributo principal puede tener valor nulo (desconocido) .

Para ilustrar el concepto de integridad referencial , añadamos a la Base de Datos de ejemplo , una relación de **Proyectos** y otra denominada **Trabajo ,** en la que se detallan los empleados que trabajan en distintos proyectos y las funciones que en ellos desempeñan .

Proyectos

P#	€	Cliente	Inicio	Fin
P10	600,000	Montegancedo SA	2003	2004-03

Trabajo.

P#	DNI	Función
P10	1023R	Director

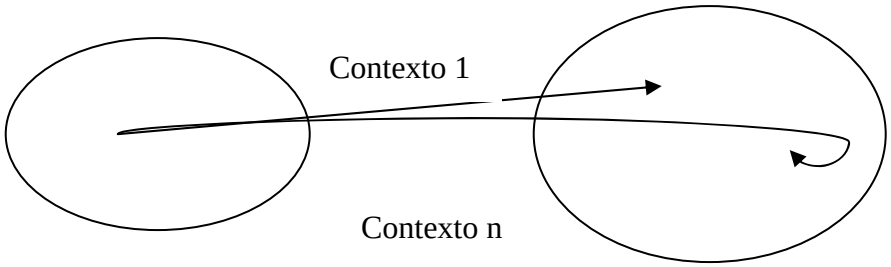
Es fácil ver que la clave de **Trabajo** es **DNI , P# .**
La restricción de referencia obliga a que si (P10,1023R) es un valor de la clave en Trabajo , P10 sea un valor de la clave en Proyectos y 1023R lo sea de DNI en Empleados .

Para concluir esta breve presentación de las restricciones y dependencias , unas notas sobre **Dependencias Multivaluadas .**

Las dependencias multivaluadas (MVD) son de tipo estructural (relativas al contexto) y por ello algo más difíciles de detectar . Si tiene lugar la mvd de Y respecto de X ,
Ello se representa como **X** $\rightarrow\!\!\rightarrow$ **Y.**
Una mvd es una correspondencia entre **Dom(X)** y Π **(Dom(Y))** , de modo que a un valor de X le corresponde un conjunto **bien definido** de valores de Y (independiente de cuáles sean los valores de los demás atributos del mismo

esquema).

Dom (X) Π **(Dom (Y))**



El objetivo de este curso es adquirir la capacidad de diseñar correctamente Bases de Datos Relacionales .
Para ello es imprescindible deducir del análisis del problema las **Dependencias funcionales** , lo que nos permitirá obtener un conjunto de **relaciones normalizadas** a partir de la **Relación Universal** (definida sobre todos los atributos del problema) . Un posterior análisis estructural de las relaciones así obtenidas , nos permitirá detectar posibles mvd's para eventualmente "refinar" el diseño hasta la denominada "Cuarta Forma Normal".
No es objetivo de este curso el estudio de las dependencias de unión ni generalizadas .

1.5 Bibliografía básica .

- **Database and Knowledge-Base Systems**
  J.D.Ullman
  Computer Science Press , 1988
- **The Theory of relational Databases**
  D.Maier
  Computer Science Press , 1983
- **Relational Databases**
  Chao-Chih Yang

Prentice Hall , 1986

Bases de Datos

- **Modelo Relacional : Algebra Relacional**

C.Fernández Baizán
Curso 2003/04

Introducción

Definición: Lenguaje de Manipulación de Datos

- **Se define así todo lenguaje que permite realizar el siguiente Conjunto de Operaciones en la BD:**
  - **Recuperación de información.**
  - **Inserción de información nueva.**
  - **Eliminación de información.**
  - **Modificación de información.**

Definición: Lenguaje de Interrogación o de Consulta

- **Subconjunto de un Lenguaje de Manipulación de Datos (DML), que sólo permite recuperación de información mediante consultas.**

Tipos de Lenguajes de Interrogación

- **Procedimentales: requieren que el usuario especifique qué datos quiere y cómo obtenerlos. El lenguaje tipo es el *álgebra relacional.***
- **No procedimentales: requieren que el usuario especifique qué datos quiere, sin especificar cómo obtenerlos. El lenguaje tipo es el *cálculo relacional* tanto el orientado a tuplas, como el orientado a dominios.**

Introducción

Limitaciones de los Lenguajes de Interrogación

- Se dice que un Lenguaje de Interrogación es completo si es al menos tan expresivo como el álgebra relacional. El Cálculo Relacional es equivalente al álgebra relacional.
  - Esta noción de completitud se refiere al poder de selección que tienen los lenguajes y no al *poder computacional* de los mismos.
  - Todos los Lenguajes de Interrogación teóricos tienen el mismo poder de selección (esto no es cierto en los DML).
- Existen consultas, que pese a tener respuesta con datos que están almacenados en la BD, no se pueden expresar:
  - Consultas a nivel de tupla que requieren ser definidas mediante expresiones aritméticas o de tratamiento de cadenas de caracteres.
  - Cálculos que incluyan conjuntos de tuplas, tales como contar el número de tuplas que cumplan una condición, hacer una media aritmética o una suma (denominadas *operaciones de agregación*).
  - Consultas que requieran un proceso de deducción.

Álgebra Relacional

Definición

- Operandos: Relaciones.
- Operadores:
  - Básicos.
  - Derivados.
- Está basada en el álgebra de conjuntos (fue definida por Codd).

Operadores Básicos

- Unión.
- Diferencia.
- Producto Cartesiano.
- Proyección.
- Selección.

Operadores Derivados

- Intersección.
- División.
- Asociación.
- Unión Natural o *Join*.

Álgebra Relacional

Definición Previa

- **Dos relaciones son unión-compatible si tienen el mismo grado y están definidas sobre los mismos atributos.**

Definición: Unión

- **La unión de dos relaciones R y S ($R \cup S$) es una relación que contiene todas las tuplas que pertenecen a R, a S o a las dos.**
- **Las relaciones R y S deben ser unión-compatible.**

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

S	A	B	C
	4	0	2
	1	6	9

R∪S	A	B	C
	3	5	8
	7	5	2
	1	6	9
	4	0	2

Álgebra Relacional

Definición: Diferencia

- La diferencia de dos relaciones R y S (R - S) es una relación que contiene todas las tuplas que pertenecen a R, pero no a S.
- Las relaciones R y S deben ser unión-compatible.

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

S	A	B	C
	4	0	2
	1	6	9

R-S	A	B	C
	3	5	8
	7	5	2

Álgebra Relacional

Definición: Producto Cartesiano

- **Sean R y S dos relaciones de grado m y n, el producto cartesiano (R × S) es una relación de grado *m*+*n* constituída por todas las tuplas en las que los *m* primeros elementos son una tupla de R y los *n* últimos una de S.**

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

S	D	E	F
	4	0	2
	1	6	9

R×S	A	B	C	D	E	F
	3	5	8	4	0	2
	3	5	8	1	6	9
	7	5	2	4	0	2
	7	5	2	1	6	9
	1	6	9	4	0	2
	1	6	9	1	6	9

Álgebra Relacional

Definición: Proyección

- **La proyección de una relación R sobre un conjunto de atributos X de T ($\Pi_X(R)$) es una relación formada por las mismas tuplas de R, pero sólo con los valores de los atributos de X.**

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

$\Pi_{BC(R)}$	B	C
	5	8
	5	2
	6	9

$\Pi_{B(R)}$	B
	5
	6

Álgebra Relacional

Definición: Selección

- **Sea una fórmula F que involucra:**
  - **Operandos: constantes y atributos.**
  - **Operadores aritméticos de comparación.**
  - **Operadores lógicos.**

  La selección de una relación R con la fórmula F ($\sigma_F(R)$) es una relación formada por el conjunto de las tuplas de R, tales que cumplen el valor de verdad de la fórmula F.

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

$\sigma_{F1}(R)$	A	B	C
	3	5	8
	1	6	9

$\sigma_{F2}(R)$	A	B	C
	3	5	8

F1: A<'5'

F2: B='5' and C>'4'

Operadores Derivados

Definición: Intersección

- La intersección de dos relaciones R y S ($R \cap S$) es una relación que contiene todas las tuplas que pertenecen a R y a S.
- Las relaciones R y S deben ser unión-compatible.

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

S	A	B	C
	4	0	2
	1	6	9

R∩S	A	B	C
	1	6	9

Operadores Derivados

Definición: División

- **Sean R_1 y R_2 relaciones de grado g_1 y g_2, respectivamente, siendo $g_1 > g_2$, $T_2 \subseteq T_1$ y $R_2 \neq \emptyset$. Definimos el cociente entre R_1 y R_2 como el conjunto de todas las (g_1-g_2)_tuplas "t" tales que para cada tupla $u \in R_2$, la tupla $tu \in R_1$. Se representa por: $R_1 \div R_2$.**

Ejemplo

R	A	B	C	D
	3	5	8	4
	7	5	2	5
	1	6	9	2
	3	5	2	5
	1	7	8	4

S	C	D
	8	4
	2	5

R÷S	A	B
	3	5

Operadores Derivados

Definición: Asociación

- **Dadas dos relaciones R_1 y R_2, de grados g_1 y g_2, respectivamente, la asociación de las mismas se representa por: $R_1 \overset{}{\underset{i\theta j}{\bowtie}} R_2$ donde:**
  - **i = número comprendido entre 1 y g_1 (o el nombre del atributo de R_1 que ocupa dicha posición),**
  - **j = número comprendido entre 1 y g_2 (o el nombre del atributo de R_2 que ocupa dicha posición),**
  - **θ = operador aritmético de comparación.**

  Se define la asociación como el conjunto de tuplas pertenecientes al producto cartesiano de R_1 y R_2, que cumplen la condición de que la columna i-ésima de R_1 está relacionada con la columna j-ésima de R_2, según el operador θ.

Operadores Derivados

Ejemplo

R	A	B	C
	3	5	8
	7	5	2
	1	6	9

S	D	E
	8	4
	7	1
	2	5

R $\overset{\theta}{\bowtie}$ S	A	B	C	D	E
	3	5	8	8	4
	3	5	8	2	5
	1	6	9	8	4
	1	6	9	2	5

$i\theta j$: A<E

Operadores Derivados

Definición: Unión Natural o *Join*

- **Sean R y S dos relaciones definidas sobre cabeceras con un descriptor común, la unión natural ($R^{\triangleright\triangleleft}S$) se calcula del modo siguiente:**
  - **Se calcula $R \times S$.**
  - **Para cada atributo de la intersección de las cabeceras, se seleccionan las filas en que coincide su valor.**
  - **Se eliminan las columnas repetidas.**

Ejemplo

R	A	B	C	D
	3	5	8	4
	7	5	2	5
	1	6	9	2
	3	5	2	5
	1	7	8	4

S	D	E
	4	7
	5	3

$R^{\triangleright\triangleleft}S$	A	B	C	D	E
	3	5	8	4	7
	1	7	8	4	7
	7	5	2	5	3
	3	5	2	5	3

Expresión de la división en términos de operadores básicos.

R

A	B	C
1	2	3
1	2	1
3	4	3
3	5	1

S

C
3
1

R:S

A	B
1	2

Sea Y=AB

$\Pi_Y(\mathbf{R})=\mathbf{R_1}$

R_1

A	B
1	2
3	4
3	5

R_1xS

A	B	C
1	2	3
1	2	1
3	4	3
3	4	1
3	5	3
3	5	1

(R_1xS)-R

A	B	C
3	4	1
3	5	3

$R_2=\Pi_Y((R_1 x S)-R)$

A	B
3	4
3	5

R_1-R_2

A	**B**
1	**2**

En general se tiene el siguiente resultado :
Sean X e Y descriptores tales que $Y \subseteq X$, siendo Z=X-Y

$R(X):S(Y)=\Pi_Z(R)-\Pi_Z((\Pi_Z(R)xS)-R)$

Estrategias de optimización

Cuando se planifican accesos en términos de operadores del Algebra Relacional , los tiempos de respuesta pueden variar significativamente con la secuencia de operaciones elegida , aun pudiendo haber varias equivalentes desde un punto de vista "matemático".
Dos son los criterios esenciales para mejorar los tiempos de ejecución/respuesta :

- Efectuar las selecciones lo antes que se pueda
- En el producto cartesiano y operaciones derivadas del mismo, el primer factor debe ser la relación con menor número de tuplas .

Ejemplo :

$\sigma_{A=1}$ (**R** $\times$ **S**)

Supongamos que **A** es un atributo tanto de **R** como de **S** , y que " a priori " podemos asegurar que el número de tuplas en **S** que satisfacen la condición de selección es menor que el correspondiente en **R** . (Ej.: En **S** se contiene la tabulación del salario en función de categoría profesional y antigüedad en la Empresa , **A** y en **R** se contienen los datos de todos los empleados).
El acceso más rápido viene dado por :
$\sigma_{A=1}$ (**S**) **x** $\sigma_{A=1}$ (**R**) .

3.Dependencias funcionales.

3.1 Definiciones básicas . Una dependencia funcional es una restricción inherente a la semántica de los atributos que se expresa en la forma : **X** $\rightarrow$ **Y (** X e Y son descriptores , esto es, conjuntos de atributos) y se lee " X implica Y" .

El significado , ya aclarado en la introducción , es que , a cada valor de X , le corresponde uno solo de Y .

Si $\neg \exists$ X'$\subset$X : X' $\rightarrow$ Y , se dice que la dependencia es **total** . En caso contrario , se dice que es **parcial** .

Si $\exists$ Z : Z $\cap$ X = Φ , Z $\cap$ Y = Φ , X $\rightarrow$ Z , $\neg$ (Z $\rightarrow$ X) , Z $\rightarrow$ Y ; de la dependencia se dice que es **transitiva .**

Un esquema de relación **R** viene definido por el par (**T , L**) , siendo **T** el conjunto de atributos y **L** el conjunto de dependencias .

Una **clave** del esquema es un descriptor **K** (subconjunto de T) , tal que :
K $\rightarrow$ **T** , siendo ésta una dependencia total .

3.2 Axiomas de Armstrong .

Los Axiomas de Armstrong son más bien reglas de inferencia .

Estas reglas permiten deducir todas las dependencias funcionales que tienen lugar entre un conjunto dado de atributos , como consecuencia de las dependencias " dato" , esto es , de las que se asumen como ciertas a partir del conocimiento del problema .

Las dependencias " dato " equivalen a los axiomas de una teoría (en sentido lógico) y los Axiomas de Armstrong son el conjunto **completo** de reglas deductivas que nos permite deducir cualquier otra dependencia **cierta** (teorema de la teoría) .

Los Axiomas de Armstrong son :

- **Reflexividad :** $\forall X , X \rightarrow X$
- **Proyectividad :** $\{ X \rightarrow Y , Z \subseteq Y \} \Rightarrow X \rightarrow Z$
- **Aumentatividad** : $\{ X \rightarrow Y , Z \supseteq X \} \Rightarrow Z \rightarrow Y$
- **Aditividad** :$\{ X \rightarrow Y , Z \rightarrow V \} \Rightarrow X \cup Z \rightarrow Y \cup V$
- **Transitividad** : $\{ X \rightarrow Z , Z \rightarrow Y \} \Rightarrow X \rightarrow Y$

El carácter correcto de estas reglas , es obvio y , en todo caso , fácilmente demostrable . Lo que ya es más complicado es demostrar el carácter completo del conjunto , demostración que pertenece al campo de la Lógica y no es en todo caso objeto de este curso básico de Bases de Datos .
Lo importante para nosotros es que este resultado (la completitud del conjunto) , nos permite abordar y resolver una serie de problemas fundamentales que luego nos conducirán al establecimiento de algoritmos de diseño sencillos y fiables .
Estos problemas fundamentales , son :

- Cierre de un conjunto de dependencias.
- Equivalencia lógica de esquemas .
- Deducción de dependencias .
- Cierre de un descriptor respecto de un conjunto de dependencias.
- .Cálculo de las claves de un esquema .

3.3 Cierre de un conjunto de dependencias . Equivalencia de esquemas .

Sea el esquema R (T , L) . El cierre del conjunto L de dependencias funcionales es el conjunto de todas las dependencias ciertas (deducibles a partir de L mediante los Axiomas de Armstrong) . El cierre se representa por $L^{+.}$
Dos conjuntos de dependencias , L y M , relativas al mismo universo T de atributos , son equivalentes si $L^{+} = M^{+}$, en cuyo caso R (T , L) y S (T , M) son esquemas equivalentes (representaciones alternativas y equivalentes del mismo problema) .

3.4 Deducción de dependencias . Cierre de un descriptor respecto de un conjunto de dependencias .

El problema de deducción puede plantearse en estos términos : Dado un esquema R (T , L) , y dada la dependencia $X \rightarrow Y$ (no perteneciente a L) , establecer el carácter cierto o falso de la misma .

La solución (teórica) es , obviamente , calcular L^+ , y ver si $X \rightarrow Y$ pertenece o no al cierre calculado .

El problema radica en la extraordinaria complejidad computacional del cálculo de un cierre (el libro de Ullmann citado en la bibliografía básica comenta casi siempre algo acerca de la complejidad computacional de los algoritmos que propone) , así que no calcularemos L^+ en ningún caso .

El cierre de un descriptor X respecto de un conjunto de dependencias L , se representa por X^+_L (se omite el subíndice si no hay ambigüedad) . El cierre se define así:

$X \rightarrow X^+ \ \wedge \neg \exists Z \supset X^+ : X \rightarrow Z$

Es decir : X implica a su cierre y no existe ningún superconjunto del mismo que sea implicado por X (el cierre , en ese sentido , es un conjunto máximo).

A diferencia del cálculo del cierre de un conjunto de dependencias , el cálculo del cierre de un descriptor es computacionalmente simple y permite resolver el problema antes planteado mediante una simple estrategia . Para saber si $X \rightarrow Y$ es cierta en relación a R(T , L) , calculamos X^+. Si $Y \subseteq X$ la dependencia es cierta y no lo es en caso contrario.

Si se trata de analizar la equivalencia de dos representaciones del mismo problema , esto es , si R(T , L) $\approx$ S(T , M) , hay que establecer la equivalencia de los conjuntos M y L de dependencias funcionales .

M y L serán equivalentes si sus cierres coinciden . $\forall (X \rightarrow Y) \in$ {M-L} se calcula el cierre X^+_L . Si Y es un subconjunto de dicho cierre en todos los casos , pasamos a calcular V^+_M , $\forall (V \rightarrow Z) \in \{ L - M \}$. Si Z es un subconjunto del cierre en todos los casos , L y M son conjuntos equivalentes .

3.5 Cálculo del cierre de un descriptor respecto de un conjunto de dependencias funcionales .

Es un proceso iterativo . Sea X el descriptor y L = { f_1 , … ,f_n } el conjunto de dependencias . Se calcula la secuencia X^0 , … , X^i , … , $X^n=X^+$ en la forma siguiente :

$X^{0=}X$

$X^{i+1}=X^i \cup$ { atributos de la derecha de las dependencias en L implicadas por subconjuntos de X^i } .

Si $X^i=X^{i+1}$, X^i es el cierre.

Si $X^i=T$, X^i es el cierre .

Ejemplo:

X=BD , f_1= AB$\rightarrow$ C , f_2=C $\rightarrow$A , f_3=BC$\rightarrow$ D , f_4= ACD$\rightarrow$ B , f_5=D$\rightarrow$ EG , f_6=CG $\rightarrow$BD , f_7=BE$\rightarrow$C , f_8= CE$\rightarrow$ AG

X^0=BD

X^1=BDEG (usando f_5)

X^2=BDEGC (usando f_7)

X^3=BDEGCA (usando f_2)

Como no hay más atributos , éste es el cierre .

3.6 Recubrimiento no redundante .

Si dos conjuntos de dependencias funcionales , L y M , son equivalentes , o sea , tienen sus cierres iguales ; se dice que L es un recubrimiento de M (y recíprocamente) .

Dado el conjunto L de dependencias funcionales , M es un recubrimiento no redundante si :

- L$\approx$ M
- Toda dependencia en M es de la forma X$\rightarrow$A (segundos miembros simples)
- Ningún implicante contiene atributos superfluos (un atributo es superfluo si su supresión no altera el cierre del conjunto) .

- No hay dependencias redundantes (una dependencia es redundante si su supresión no altera el cierre del conjunto) .

Ejemplo :

Calcular un recubrimiento no redundante de :

L={ $AB \rightarrow C$, $C \rightarrow A$, $BC \rightarrow D$, $ACD \rightarrow B$, $D \rightarrow EG$, $BE \rightarrow C$, $CG \rightarrow BD$, $CE \rightarrow AG$ }

Aplicando el axioma de proyectividad :

L_1={ $AB \rightarrow C$, $C \rightarrow A$, $BC \rightarrow D$, $ACD \rightarrow B$, $D \rightarrow E$, $D \rightarrow G$, $BE \rightarrow C$, $CG \rightarrow B$, $CG \rightarrow D$, $CE \rightarrow A$, $CE \rightarrow G$ }

Cuando hay redundancies obvias , pueden (y deben) ser eliminadas cuanto antes , pues esto contribuirá a simplificar el problema en su conjunto .

Un rápido examen, permite observar que , ya que $C \rightarrow A$, $CE \rightarrow A$ es redundante y puede eliminarse . Asimismo , $ACD \rightarrow B$ puede sustituirse por $CD \rightarrow B$.

Nos queda el conjunto L_2 en el que , por comodidad de notación , vamos a identificar cada elemento mediante un número de orden :

L_2= { $AB \rightarrow C$ (1) , $C \rightarrow A$ (2) , $BC \rightarrow D$ (3) , $CD \rightarrow B$ (4) , $D \rightarrow E$ (5) , $D \rightarrow G$ (6) , $BE \rightarrow C$ (7) , $CG \rightarrow B$ (8) , $CG \rightarrow D$ (9) , $CE \rightarrow G$ (10) }

Eliminación de atributos superfluos:

Si una dependencia viene implicada por un atributo simple , es obvio que éste no es superfluo .

Analicemos los implicantes compuestos : $AB \rightarrow C$. Si A fuese superfluo , B no podrá serlo , y recíprocamente .

Supongamos que se elimina A . Nos queda $B \rightarrow C$. Si es cierta , A es superfluo

Calculamos B^+_{L2} y se obtiene B *A no es superfluo*.

Si eliminamos B ,nos queda $A \rightarrow C$. El cierre A^+_{L2} es A . *B no es superfluo.*

Repitiendo este proceso para todos los implicantes compuestos , puede verse que no hay atributos superfluos *(y es un ejercicio muy recomendable para fijar los conceptos)*.

Eliminación de dependencias redundantes :

Podemos seguir el orden que queramos , llegando eventualmente a recubrimientos no redundantes distintos , pero equivalentes .
Para ver si (1) es redundante , la elimino y calculo $AB^{+}_{L2-\{1\}}$:

$AB^{+}_{L2-\{1\}}$=AB . Como C no pertenece al cierre , (1) no es redundante .
Repetimos el proceso:
$C^{+}_{L2-\{2\}}$= C , así que (2) no es redundante .
$BC^{+}_{L2-\{3\}}$=ABC , así que (3) no es redundante .
$CD^{+}_{L2-\{4\}}$=CDAEGB , así que (4) es redundante y se suprime , siendo $L_3=L_2-\{4\}$
$D^{+}_{L3-\{5\}}$=DG , así que (5) no es redundante .
$D^{+}_{L3-\{6\}}$=DE , así que (6) no es redundante .
$BE^{+}_{L3-\{7\}}$=BE , así que (7) no es redundante .
$CG^{+}_{L3-\{8\}}$=CGAD , así que (8) no es redundante .
$CG^{+}_{L3-\{9\}}$=CGABD , así que (9) es redundante y se suprime , siendo $L_4=L_3-\{9\}$
$CE^{+}_{L4-\{10\}}$=CEA , así que (10) no es redundante .

Un recubrimiento no redundante de L es , por tanto :

$AB \rightarrow C$; $C \rightarrow A$; $BC \rightarrow D$; $D \rightarrow E$; $D \rightarrow G$; $BE \rightarrow C$; $CG \rightarrow B$; $CE \rightarrow G$

En problemas de diseño y cálculo de claves , trabajaremos siempre con recubrimientos no redundantes .

3.7 Determinación de las claves de un esquema .

Ahora ya podemos formular la base del procedimiento . Sea el esquema R (T , L) con T un conjunto de n atributos .
Para calcular las claves , vamos generando los posibles descriptores a partir de los elementos de T, y vamos calculando los cierres respecto de L . Si X^{+}_{L}=T, X es clave (y ya ignoramos sus superconjuntos) .

Lo malo es que el número de descriptores a explorar (en el caso peor) es 2^n, y es que el problema de calcular todas las claves de un esquema es del tipo NP-completo , por lo que todos los algoritmos existentes para su computación requieren de mucho tiempo para su ejecución .

Afortunadamente , los problemas de diseño sólo requieren del conocimiento de al menos una clave , y , determinar una clave , no suele ser complicado .

En este curso , y por limitación del tiempo disponible para impartir este tema , nos limitaremos a calcular una clave de R (T , L) para luego "rediseñar" el esquema en forma de proyecciones normalizadas .

3.8 Cálculo de una clave del esquema R (T , L) .

Lo primero que debe hacerse , es calcular un recubrimiento no redundante de L .Esto "despeja" un poco los datos de partida y facilita el cálculo de la(s) clave(s) , sea cual sea el procedimiento empleado .

Ya hemos dicho que el cálculo de todas las claves no es una necesidad en la práctica del diseño . Sólo tendría un sentido si lo que se pretende es establecer el nivel de normalización de un esquema (enfoque analítico) , para validar un diseño.

Pero , ya que partimos de algoritmos que aseguran la normalización del resultado , no tiene más interés en este curso .

Ahora bien , para diseñar correctamente , necesitamos conocer al menos una clave . Afortunadamente , esto no es muy complicado .

Sea L un conjunto no redundante de dependencias funcionales . Distinguiremos cuatro conjuntos de atributos :

I={ atributos que aparecen sólo como implicantes }

D={ atributos que aparecen sólo como implicados }

ID={ atributos que aparecen como implicantes e implicados }

N={ atributos que no aparecen en ninguna dependencia }

Es fácil razonar lo siguiente :

- Los atributos de **I** forman parte de todas las claves .
- Los atributos de **N** forman parte de todas las claves .
- Los atributos de **D** no forman parte de ninguna clave .
- Los atributos de **ID** pueden formar parte o no de alguna clave .

Un procedimiento sencillo para determinar una clave , es por tanto el siguiente :

1. Calcular $Z=I \cup N$. Z es el núcleo del proceso (intersección de todas las claves , pudiendo ser en algún caso el conjunto vacío)
2. Calcular Z^+ . Si este cierre es T , Z es la clave (única) del esquema. Si no es así , sigue la búsqueda de una clave .
3. $Z^0=Z$, $Z^i=Z^{i-1} \cup \{ A_j \}$, $A_j \in ID$
4. Calcular $(Z^i)^+$. Si este cierre es T , Z^i es , en general una superclave del esquema (es clave o contiene alguna clave)
5. La(s) clave(s) contenida(s) en Z^i se extrae(n) por eliminación de posibles atributos superfluos (distintos de los del núcleo y del último añadido) .

En el ejemplo del apartado anterior para cálculo de un recubrimiento no redundante , es fácil ver :

ID ={ A B C D E G }

I=ϕ , N=ϕ , D= ϕ

$Z^0 = \phi$.

Z^1=A , A^+=A

Z^2=AB , AB^+=T

AB es una clave .

(Nota:

Con relación al punto 3,en el que el núcleo se va incrementando con atributos del conjunto **ID**, si se diese el caso de que el cierre de Z^i incluyese elementos de **ID**, es obvio que será innecesario añadirlos.

Esto es una heurística que puede agilizar el proceso).

4.Diseño de Bases de Datos (I)

4.1 Anomalías .

Se denominan así en teoría de Bases de Datos a ciertos problemas que aparecen con frecuencia en el manejo de las mismas cuando el diseño no ha sido realizado de forma “normalizada” (en este tema se aclara el significado del término “normalización “ y se introducen algoritmos de síntesis normalizada de relaciones , teniendo en cuenta las dependencias funcionales existentes).

Distinguiremos tres “anomalías” básicas :

- Anomalía de inserción : Imposibilidad de dar de alta una tupla por no disponer del valor de un atributo principal .
- Anomalía de borrado : Pérdida de información por dar de baja una tupla.
- Anomalía de modificación : Tiene que ver con la redundancia (repetición de la misma información en tuplas diferentes y consiguiente necesidad de propagar actualizaciones) . En general , la normalización reduce la redundancia , pero no la elimina por completo .

Ejemplo:

Proveedores

P#	**A#**	**€**	**C**	**P**
P1	A21	125	Alcorcón	Madrid
P1	A06	85	Alcorcón	Madrid

En la anterior relación , el significado de los atributos es : Código de proveedor , código de artículo , precio del artículo , ciudad en la que el proveedor tiene su domicilio y provincia en la que el proveedor tiene su domicilio .
Suponemos que el domicilio de un proveedor es único y que el precio de un artículo no depende del proveedor .
Las dependencias funcionales asociadas a este esquema , son :
$P\# \rightarrow C$
$P\# \rightarrow P$
$C \rightarrow P$
$A\# \rightarrow €$

Y la clave es P#A#

Veamos ahora qué problemas pueden presentarse en el manejo de esta relación tan simple .
Antes de nada , diremos que una relación está en Primera Forma Normal (en lo sucesivo , 1FN) , si todas sus entradas son simples .

Codd definió las Formas Normales Segunda (2FN) y Tercera (3FN) en la forma siguiente :

- Una relación 1FN está en Segunda Forma Normal si ningún atributo no principal depende parcialmente de ninguna clave .
- Una relación 2FN está en tercera Forma Normal si ningún atributo no principal depende transitivamente de ninguna clave .

Volvamos a nuestro ejemplo . Si en **Proveedores** queremos dar de alta un nuevo artículo , A25, de precio 75 ; pero aún no hemos decidido el proveedor ; no podremos hacerlo , ya que el código de proveedor forma parte de la clave . Este es un ejemplo de anomalía de inserción , producido por la dependencia parcial del precio respecto de la clave .

Parece lógico separar la información propia de proveedores por un lado y de artículos por otro , manteniendo además el nexo entre ambos en una nueva relación .
Un nuevo diseño mejorado es por tanto el siguiente :

Proveedores

P#	**C**	**P**
P1	Alcorcón	Madrid

Artículos

A#	**€**
A21	125
A06	85
A25	75

Suministra

P#	**A#**
P1	A21
P1	A06

Las tres relaciones son 2FN .

Supongamos ahora que P1 es el único proveedor domiciliado en Alcorcón . Si causa baja en nuestra Base , perdemos la información de que Alcorcón está en Madrid .
Este es un caso de anomalía de borrado , y se debe a la dependencia transitiva de provincia respecto de proveedor .

La eliminación de esta dependencia transitiva , lleva a sustituir la primera relación por otras dos :

Proveedores

P#	C
P1	Alcorcón

Localización

C	P
Alcorcón	Madrid

El conjunto **{ Proveedores , Artículos , Suministra , Localización }** está en 3FN .
Se han eliminado las anomalías de inserción y borrado y disminuído las de actualización .

4.2 Definición formal del proceso de síntesis (diseño) de relaciones normalizadas.

La normalización o síntesis de esquemas relacionales normalizados es en esencia la sustitución de un esquema R (T , L) por un conjunto de proyecciones { R_1 (T_1 , L_1) , …… , R_n (T_n , L_n) } que represente la misma información y tenga mejores propiedades de manejo (eliminación de anomalías) .

Es importante tener en cuenta que L_i es lo que se denomina " proyección de L sobre T_i " , con el siguiente significado :
$Li=\{ X \rightarrow Y \in L^+ / (X \cup Y) \subseteq Ti \}$

Las condiciones que debe cumplir el diseño , son :

- Esquemas resultantes normalizados.
- Conservación de atributos : $\bigcup_i^n{}_{=1} T_i = T$
- Conservación de información (tuplas) : $IXI_i^n{}_{=1} R_i = R$ (carácter "lossless-join")
- Conservación de dependencias : $(L_1 \cup ..... \cup L_n)^+ = L^+$

En general , la unión natural de las proyecciones es un superconjunto de la relación original , o sea : Al realizar la unión natural para resolver determinadas interrogaciones a la Base , el resultado contendrá las respuestas correctas y otras falsas (generadas por el propio proceso de "join") , que no son discernibles de aquéllas .

El Algoritmo de Ullmann (ver Bibliografía señalada en el Capítulo de Introducción) , permite obtener en una forma muy simple un conjunto de esquemas 3FN a partir de otro dado , cumpliendo las cuatro condiciones antes señaladas .

4.3 Algoritmo de Ullmann

Sea el esquema R (T , L) , donde suponemos L no redundante (si no lo fuese , calculamos un recubrimiento no redundante , con el cual trabajaremos) .

Agrupamos las dependencias en L con el mismo implicante . A cada grupo , se le asocia un esquema .
Si en ningún esquema aparece una clave de R , añadimos un esquema adicional sobre una clave (basta con conocer sólo una) .

El resultado es un conjunto 3FN , "lossless-join" y que preserva dependencias .

En el anterior apartado usamos como ejemplo
Proveedores (T , L)
T= { P# , A#, €, C , P }
L={ P# $\rightarrow$ C, A#$\rightarrow$€ , C$\rightarrow$ P }
Clave: P# A#

La aplicación del Algoritmo de Ullmann produce como resultado :

R_1(P# , C) , R_2 (A# , €) , R_3 (C , P) , R_0 (P# , A #)

R_1|x| R_2|x| R_3|x| R_0 = R

Es obvia la preservación de dependencias .

4.4 Teorema de Delobel y Casey .

Sea R_1 la proyección sobre T_1 de R (T, L) y sea R_2 la proyección del mismo sobre T_2 , siendo $T_1 \cup T_2$=T , $T_1 \cap T_2 \neq \phi$.

La unión natural ("join") de ambas proyecciones es , en general , un superconjunto de R :

$$R_1 |x| R_2 \supseteq R$$

Ya hemos comentado que esto puede dar lugar a la obtención (no detectable) de respuestas erróneas .

El Teorema de Delobel y Casey (cuya forma general veremos en el tema de dependencias multivaluadas) , establece una condición <u>suficiente</u> para que la unión natural de R_1 y R_2 sea <u>exactamente</u> R :

$$\{ T_1 \cap T_2 \rightarrow T_1\text{-}T_2 \in L^+ \vee T_1 \cap T_2 \rightarrow T_2\text{-}T_1 \in L^+ \} \Rightarrow R_1 |x| R_2 = R$$

El Algoritmo de Ullmann garantiza que el "join" de <u>todas</u> las relaciones sintetizadas es igual a la relación original (universal) , pero el "join" es una operación de alto coste en términos de tiempo de ejecución , por lo que deberíamos intentar no realizar "joins" innecesarios .

Supongamos que { R_1(A, B , C) , R_2 (C , D) , R_3 (D , E) } es el resultado de aplicar a R ({ A , B , C , D } , L) el Algoritmo de Ullmann .
Sea la interrogación : " Valores de C para los cuales E= 0 ".
La formulación del acceso correspondiente en la forma $\Pi_C(\sigma_{E=0}(R_3)|x|R_2)$, será correcta si coincide con $\Pi_C(\sigma_{E=0}(R)$) , lo que sólo podemos garantizar si tiene lugar la condición suficiente descrita en el Teorema de Delobel y Casey :

$D \rightarrow C$ o bien $D \rightarrow E$ es cierta (al menos una de ellas pertenece al cierre L^+) .

4.5 Forma Normal de Boyce-Codd (BCFN)

Un esquema 1FN , está en BCFN si toda dependencia no trivial está implicada por una clave ($X \rightarrow Y$ es trivial si $X \supseteq Y$) .

Toda relación BCFN es también 3FN . El recíproco , no es cierto :

BCFN $\Rightarrow$ **3FN** , **3FN** $\neg \Rightarrow$ **BCFN**

Todo esquema admite una descomposición BCFN "lossless-join" , pero no siempre es posible que el proceso conserve las dependencias .
Vamos a ilustrar estas ideas con un ejemplo sencillo :

R (T , L)
T = { A , B , C }
L = { $AB \rightarrow C$, $C \rightarrow B$ }

El conjunto L es no redundante y la única clave es AB .
Existe una dependencia no trivial , $C \rightarrow B$, cuyo implicante no es clave .

Un diseño BCFN y "lossless-join" debe necesariamente aislar en un esquema a C y B y, por otro lado , teniendo en cuenta el teorema de Delobel y Casey , otro esquema contendrá los atributos A y C .

El resultado es , por tanto :

R_1 (A , C) ; $L_1=\Phi$

R_2 (B , C) ; $L_2=C \rightarrow B$

$(L_1 \cup L_2)^+ \subset L^+$ (la dependencia de C respecto de AB se ha perdido) .

$R_1|x|R_2$ = R , ya que $T_1 \cap T_2 \rightarrow T_2-T_1$

En general , el proceso de síntesis BCFN es el siguiente :

Partimos de R (T , L) , siendo L no redundante .

1. Se calculan <u>todas</u> las claves de R. Si no existe ningún implicante no clave , el proceso concluye . **R es BCFN .**
2. Si existe $X \rightarrow Y$ con X no clave , se proyecta R en R_1 ($X \cup Y$) y en R_2(T –Y) . **R_1 es BCFN .** En cuanto a R_2 , debemos calcular L_2 y las nuevas claves . Para lo primero , hay que calcular L^+ , seleccionar de ese conjunto las dependencias definidas <u>sólo sobre atributos de T_2</u> y luego eliminar redundancias . Se calculan a continuación todas las claves del esquema y se procede como en el punto anterior .

El proceso procede por tanto por sucesivas descomposiciones binarias que usan como “pivote” dependencias que vulneran la definición de BCFN , asegurando al mismo tiempo , como consecuencia del Teorema de Delobel y Casey , el carácter “lossless-join” del resultado ; no sólo considerado globalmente (como sucede en el Algoritmo de Ullmann) , sino también para el caso de efectuar el “join” de sólo algunas de las proyecciones resultantes .

El inconveniente del método es su excesiva complejidad computacional (es del tipo NP-completo) , así como el hecho de no garantizar la conservación de las dependencias .

5. Dependencias multivaluadas.

5.1 Definiciones básicas .

Una dependencia multivaluada es una sentencia que se escribe : $X \rightarrow\rightarrow Y$ y cuyo significado intuitivo es el siguiente : A cada valor de X se le asocia un conjunto de valores de Y independiente del contexto (si X e Y son subconjuntos de T , el contexto es $Z=T-(X \cup Y)$) .
Como vimos en la Introducción , esto quiere decir que la dependencia multivaluada asigna a cada valor de Dom (X) un valor de P (Dom (Y)) , y esta asignación no varía con el contexto .
Es muy importante no confundir la existencia de una dependencia multivaluada (en adelante , mvd) con el hecho de que entre los dominios de X e Y pueda establecerse una correspondencia 1 : n , lo que es un error bastante frecuente .

Veremos ahora tres definiciones (por supuesto , equivalentes) de mvd . Cada una tiene su mayor o menor utilidad según el punto de vista bajo el que las mvd's sean consideradas .

Definición usando la notación de dependencia generalizada :

X	**Y**	**Z**
x	y	z
x	y'	z'
x	**y**	**z'**
x	**y'**	**z**

$X \rightarrow\rightarrow Y$ sii la existencia de las tuplas < x y z > , <x y' z'> , implica la existencia de estas otras : <x y z'> , <x y' z'> .

En notación de dependencia generalizada , a las dos primeras se les llama "tuplas hipótesis " y a las dos últimas " tuplas conclusión " .

La consistencia requiere de la aparición de las tuplas conclusión en cualquier instancia válida del esquema en el que la mvd tiene lugar , dada la aparición de las tuplas hipótesis .
En un contexto deductivo , es fácil formular la mvd como una regla , y no es necesario que las conclusiones acompañen de modo explícito a las hipótesis ,siendo suficiente que se puedan deducir de las mismas .

Definición usando el criterio de proyección-join :

En el esquema R(T , L) tiene lugar la mvd $X \rightarrow\rightarrow Y$, sii para toda instancia r del mismo se cumple :

$\Pi_{XY}(r) \; |X| \; \Pi_{XZ}(r) = r$

Esta definición no es más que enunciar de otra forma el Teorema de Delobel y Casey que veremos enseguida .
Su utilidad es la aplicación al diseño del conocimiento previo de las mvd's , pero no resulta adecuada para la extracción de las mismas .

Definición usando el criterio de selección-proyección :

Cuando veamos la estructura inferencial de las mvd's , justificaremos que , en toda mvd , puede suponerse vacía la intersección de implicante e implicado .
Sea $Z=T-(X \cup Y)$.
La dependencia $X \rightarrow\rightarrow Y$ tiene lugar en R (T) sii , para toda instancia r del esquema :

$\Pi_Y(\sigma_{X=x} (r)) = \Pi_Y(\sigma_{X=x \wedge Z=z}(r))$, siendo x , z valores genéricos de los dominios de X y Z , respectivamente .

Sea la relación :

R (Pintor , Cuadro , Museo)
Con las dependencias funcionales :

Cuadro $\rightarrow$ Pintor
Cuadro $\rightarrow$ Museo

Dado que un pintor determinado tiene asociado el conjunto bien definido de sus cuadros , podría pensarse en si tiene o no lugar la mvd Pintor $\rightarrow\rightarrow$ Cuadro .

Designemos por comodidad a los atributos por sus iniciales . De acuerdo con la definición , la mvd tendrá lugar sii se cumple :

$\Pi_C(\sigma_{P=Goya} (R)) = \Pi_C(\sigma_{P=Goya \wedge M=Prado} (R))$

Y esto para cualquier par de valores (p , m) de los dominios de P y M .
Claramente se ve que no se cumple la igualdad con carácter general . La primera proyección nos da los cuadros de Goya , mientras que la segunda nos da el subconjunto de los mismos perteneciente a los fondos del Prado .
Esta definición es la que usaremos para extraer las mvd`s de un cierto esquema .

5.2 Axiomas para mvd's .

La estructura inferencial completa para mvd's fue enunciada por Beeri , Fagin y Howard , y puede consultarse en el libro de Ullmann citado en la Bibliografía básica .
No usaremos en este curso todos los Axiomas , ya que , siendo nuestro objetivo el diseño , y siendo las mvd's propiedades relativas al contexto ,

“aparecen y desaparecen “ en el proceso de normalización , por lo que no analizaremos todas las existentes en la relación universal R (T , L) , sino que será suficiente con ir detectando alguna de las posibles mvd’s que vulneren la definición de Cuarta Forma Normal .

Axiomas :

- Toda dependencia funcional es un caso particular de mvd .(1)
- Axiomas de Armstrong para dependencias funcionales .(2)
- Siendo Z=T-(X $\cup$ Y) , { X$\rightarrow\rightarrow$Y }$\Rightarrow$X$\rightarrow\rightarrow$Z (3)
- { X$\rightarrow\rightarrow$Y , Y$\rightarrow\rightarrow$Z } $\Rightarrow$ X$\rightarrow\rightarrow$(Z – Y) (4)
- { X$\rightarrow\rightarrow$Y } $\Rightarrow$ X$\rightarrow\rightarrow$(Y – X) (5)

Con este subconjunto de axiomas tendremos suficiente para nuestros objetivos de diseño .
Mencionar por último que una mvd es “trivial” si la unión de implicante e implicado es T .

Supongamos la siguiente relación :

Alumno	**Asignatura**	**Actividad**
J.Fernández	Inglés	Deportes
J.Fernández	Historia	Teatro
J.Fernández	Literatura	Deportes
J.Fernández	Inglés	Teatro
J.Fernández	Historia	Deportes
J.Fernández	Literatura	Teatro

En ella se reflejan las asignaturas en las que están inscritos los alumnos de un colegio y las diferentes actividades opcionales que han elegido .

El alumno J.Fernández cursa tres asignaturas y ha elegido dos actividades . Para que esta información sea correctamente reflejada en la Base de Datos , necesito seis tuplas (2 x 3) . Y ello para que la respuesta a cualquier interrogación que relacione **asignatura** con **actividad** sea **completa** .

Por lo demás , es fácil ver que la relación es 3FN , siendo su clave el conjunto de los tres atributos (no hay dependencias funcionales) .

Analicemos ahora la situación desde el punto de vista de las dependencias multivaluadas .
Por comodidad , llamaremos A al alumno , S a la asignatura y C a la actividad .
El análisis de las posibles mvd’s es sencillo .
Comenzaremos por establecer si A$\rightarrow\rightarrow$S , aplicando el criterio de selección-proyección .

$\Pi_S(\sigma_{A=a} (R))$= conjunto de asignaturas que cursa el alumno “a”.

$\Pi_S(\sigma_{A=a \wedge C=c} (R))$=conjunto de asignaturas que cursa el alumno “a” que ha elegido la actividad “c” .

El segundo acceso está “sobreespecificado” , y ambos conjuntos son idénticos .La dependencia analizada es cierta , y por tanto lo es también $A \rightarrow\rightarrow C$, como puede comprobarse aplicando el axioma (3) .

Analicemos las restantes posibilidades :

$C \rightarrow\rightarrow A$ ¿?

$\Pi_A(\sigma_{C=c} (R))$= conjunto de alumnos que han elegido la actividad “c”.

$\Pi_A(\sigma_{C=c \wedge S=s} (R))$ = conjunto de alumnos que han elegido la actividad “c” y están inscritos en la asignatura “s”.

En general , el segundo conjunto está incluído en el primero , así que la mvd no tiene lugar y , como consecuencia , tampoco $C \rightarrow\rightarrow S$ tiene lugar .

De modo análogo se llega a que no hay mvd’s no triviales implicadas por S .

No analizamos las posibles mvd’s implicadas por dos atributos , ya que , en este contexto , son triviales (una mvd es trivial si la unión de implicante e implicado es T) .

La clave del esquema R (sigue definiéndose como siempre , en relación a las dependencias funcionales) , es ASC .

La **Cuarta Forma Normal (4FN)** , fue definida por R.Fagin y es como sigue :

Un esquema 1FN es 4FN sii toda mvd no trivial está implicada por una clave .

En nuestro ejemplo , hay dos mvd’s no triviales implicadas por A , que no es clave . No es 4FN y , como hemos visto , presenta un alto grado de redundancia .

5.3 Teorema de Delobel y Casey .

Sea R(T) un esquema y sean $R_1(T_1)$ y $R_2(T_2)$ sus proyecciones sobre T_1 y T_2 , respectivamente .
La descomposición es LJ sii $T_1 \cap T_2 \rightarrow\rightarrow T_i - T_j$, (i,j)=(1,2) ó (2,1) .

5.4 Normalización 4FN .

Dado R (T , L) , una descomposición 4FN LJ se consigue en la forma siguiente :

- Calcular todas las claves.
- Analizar la existencia de posibles mvd’s
- Si se detecta alguna no trivial ni implicada por una clave , $X \rightarrow\rightarrow Y$
- Proyectar R sobre $X \cup Y$ y sobre T- Y . De acuerdo con el Teorema de Delobel y Casey , la descomposición es LJ . El caso más general , obliga a estudiar desde el principio ambas proyecciones, calculando para

ambas la proyección de L , las claves y las mvd's que pudiesen surgir en los nuevos contextos (y que , con relación al esquema inicial , se denominan "embebidas").

- Si en alguno de los subesquemas se detectan mvd's no trviales ni implicadas por alguna de las claves respectivas , el proceso continúa .

4FN $\Rightarrow$ BCFN $\Rightarrow$ 3FN

En nuestro ejemplo , proyectaríamos R sobre AS y AC , obteniendo R_1 y R_2 tales que $R=R_1 |x| R_2$ para cualquier instancia del esquema .

5.5 Ejemplo de diseño 4FN .

Consideremos la siguiente relación :

R

Asignatura	**Prerrequisito**	**Alumno**	**Año**
Ecuaciones diferenciales	Cálculo	J.Fernández	01
Ecuaciones diferenciales	Análisis matemático	J.Fernández	02
Ecuaciones diferenciales	Cálculo	E.Ruiz	00
Ecuaciones diferenciales	Análisis matemático	E.Ruiz	03
Ecuaciones diferenciales	Cálculo	S.García	01
Ecuaciones diferenciales	Análisis matemático	S.García	02

En la que , por cada asignatura que se imparte en una cierta Facultad , se indica cuáles son las que constituyen un prerrequisito para dicha asignatura , qué alumnos están matriculados en esa asignatura y en qué año aprobaron cada uno de los prerrequisitos de la misma .

Designaremos los atributos :

Asignatura (A)
Prerrequisito (B)
Alumno (C)
Año (D)

Es inmediato comprobar la existencia de una sola dependencia funcional no trivial :

$BC \rightarrow D$

La clave es ABC , de modo que R no es 2FN (D depende parcialmente de la clave) .

Para abordar el diseño 4FN , hay que analizar posibles mvd`s .

Sólo interesan las que involucran un número de atributos menor o igual que 3 , ya que , por el axioma (5) , siempre podemos trabajar con dependencias que tengan disjuntos sus dos términos , y entonces toda dependencia de 4 atributos es trivial en este contexto (la unión de sus dos términos es T) .

Diseño:

1. Usando el axioma (1) , vemos en primer lugar que tiene lugar la mvd : $BC \rightarrow\rightarrow D$
2. Aplicando el Teorema de Delobel y Casey , proyectamos sobre ABC y BCD . El esquema R_1(ABC) no tiene dependencias funcionales , por lo que su clave es ABC . En R_2 (BCD) se tiene que $BC \rightarrow D$ y la clave es BC. Ahora buscamos posibles mvd`s .
3. En R_1 , analizamos si $A \rightarrow\rightarrow B$ (si es así , también se daría $A \rightarrow\rightarrow C$, por (3) . Empleamos el criterio de selección –proyección : $\Pi_B((\sigma_{A=a}((R))$=conjunto de prerrequisitos de la asignatura“a”. $\Pi_B(\sigma_{A=a \wedge C=c}(R))$=conjunto de prerrequisitos de la asignatura “a” en la que está matriculado el alumno “c” (acceso sobreespecificado). Es obvio que ambos conjuntos coinciden por lo que $A \rightarrow\rightarrow B$ y $A \rightarrow\rightarrow C$. Esto basta para ver que R_1 no es 4FN .
4. Aplicando de nuevo Delobel y Casey , proyectamos R_1 en R_{11}(AB) y en R_{12}(AC) , ambas 4FN .
5. Analicemos ahora R_2. Veamos , para empezar , si $B \rightarrow\rightarrow C$, por aplicación del criterio de selección-proyeción : $\Pi_C(\sigma_{B=b}($ R $))$=conjunto de alumnos que han aprobado el prerrequisito “b”. El conjunto de comparación es $\Pi_C(\sigma_{B=b \wedge D=d}$ (R $))$=conjunto de alumnos que han aprobado el prerrequisito “b” el año “d”.En general ambos conjuntos son diferentes, no teniendo lugar la mvd . Puede comprobarse que no hay mvd`s no triviales en R_2 , que es por tanto 4FN (y , en consecuencia , 3FN y BCFN).

En la práctica , cuando añadamos el R_0 (clave) a nuestro diseño según Ullmann , y cuando la clave tenga 3 ó más atributos , será frecuentemente necesario pasar R_0 a 4FN .

El proceso seguido , garantiza la preservación del carácter LJ del conjunto .